(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平6－202875

(43)公開日　平成６年(1994)７月22日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/45					
		9292－5Ｂ	Ｇ０６Ｆ　9/44	３２２　Ｆ	

審査請求　有　　請求項の数５　（全　８　頁）

(21)出願番号　特願平4－360619

(22)出願日　平成４年(1992)12月28日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　山本　久仁子
東京都港区芝五丁目７番１号　日本電気株式会社内

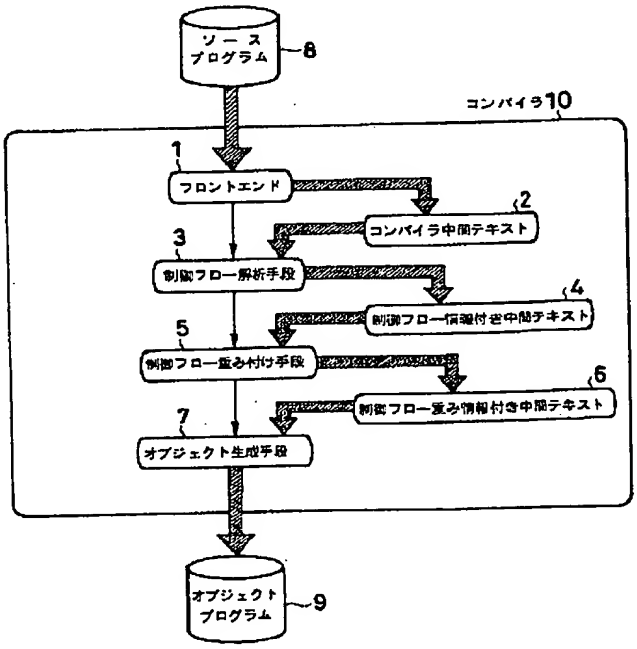
(74)代理人　弁理士　河原　純一

(54)【発明の名称】　インライン展開による最適化を行うコンパイラ

(57)【要約】

【目的】　関数呼出し毎のインライン展開の要否の決定を可能とし、実行イメージに合ったインライン展開を行い、実用的かつ効率的なインライン展開を可能とする。

【構成】　制御フロー重み付け手段５は、制御フロー解析手段３の解析結果に基づいて制御フローを構成する各部分制御フローの実行回数を推測し、その実行回数に基づいて各部分制御フローに対する重み付けを行う。オブジェクト生成手段７は、制御フロー重み付け手段５による重み付けの結果を参照して各関数呼出しによって呼び出される関数のインライン展開の要否を決定し、その決定を反映したオブジェクトプログラム９を生成する。



1

【特許請求の範囲】

【請求項１】　制御フロー解析手段の解析結果に基づいて制御フローを構成する各部分制御フローの実行回数を推測し、その実行回数に基づいて各部分制御フローに対する重み付けを行う制御フロー重み付け手段と、
この制御フロー重み付け手段による重み付けの結果を参照して各関数呼出しによって呼び出される関数のインライン展開の要否を決定し、その決定を反映したオブジェクトプログラムを生成するオブジェクト生成手段とを有することを特徴とするインライン展開による最適化を行うコンパイラ。

【請求項２】　重み付けの対象の部分制御フローに関与するループについてのループ回数およびネスト数を使って当該部分制御フローの実行回数を推測し、ループをキーとする重み付けを当該部分制御フローに対して行う前記制御フロー重み付け手段を有することを特徴とする請求項１記載のインライン展開による最適化を行うコンパイラ。

【請求項３】　処理対象の関数呼出しを含む部分制御フローに対する前記制御フロー重み付け手段による重み付けの結果と当該関数呼出しによって呼び出される関数のオブジェクトサイズとを使った算出式を用いてインライン展開判定値を算出し、そのインライン展開判定値とインライン展開要否指標値との比較に基づいて当該関数呼出しによって呼び出される関数のインライン展開の要否を決定する前記オブジェクト生成手段を有することを特徴とする請求項１記載のインライン展開による最適化を行うコンパイラ。

【請求項４】　重み付けの対象の部分制御フローに関与するループについてのループ回数が静的に決まらない場合に使用されるデフォルトループ回数をオプション情報に基づいて取得する前記制御フロー重み付け手段を有することを特徴とする請求項２記載のインライン展開による最適化を行うコンパイラ。

【請求項５】インライン展開要否指標値をオプション情報に基づいて取得する前記オブジェクト生成手段を有することを特徴とする請求項３記載のインライン展開による最適化を行うコンパイラ。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、コンパイラに関し、特に最適化の一手法である「関数のインライン展開」を行うコンパイラ（インライン展開による最適化を行うコンパイラ）に関する。

【０００２】

【従来の技術】従来、インライン展開による最適化を行うコンパイラは、以下のようにしてインライン展開を行うべきか否か（インライン展開の要否）を決定していた。

【０００３】○　一の手法では、インライン展開の要否

2

の指示が他の種類の最適化の要否の指示とともに最適化オプション（オブジェクトプログラムの実行効率をどこまで向上させるかに基づく最適化オプション）によって指定され、その最適化オプションに基づいてインライン展開の要否が決定される。この最適化オプションの指定は、プログラム全体に対して行われる（関数呼出し毎に行われることはない）。最適化オプションには、最適化レベルを指定することができるものが多い。

【０００４】①　他の手法では、コンパイル時にインライン展開のみについての要否の指示が所定のオプションによって指定され、そのオプションに基づいてインライン展開の要否が決定される。この場合にも、このオプションの指定は、プログラム全体に対して行われ、関数呼出し毎に行われることはない。

【０００５】なお、プログラム全体に対してインライン展開を行うべきことが指定された場合には、各関数のインライン展開を行うべきか否かがオブジェクトプログラムのレベルでの当該関数の大きさ（当該関数のオブジェクトサイズ）によって決定されていた。

【０００６】上述のように、従来の技術では、インライン展開の要否の決定は、関数呼出し毎ではなくプログラム全体を対象として行われており（インライン展開が画一的に適用されており）、関数のオブジェクトサイズという要因しか考慮されていなかった（関数の呼出し回数等の他の要因は全く考慮されていなかった）。

【０００７】また、上述した従来の技術に対する新技術として公開されているものには、以下に示す技術がある。以下、上述した従来の技術を「旧従来技術」といい、以下に示す技術を「新従来技術」という。

【０００８】○　「特開平３－９９３３０（手続きインライン展開方式）」に係る特許出願によって示される新従来技術：中間テキストのレベルでカウントされた関数の静的な参照回数（中間テキストに出現する当該関数の参照回数）を使用して、その関数のインライン展開の要否を決定する。

【０００９】①　「特開平１－１１８９３１（プログラム変換方式）」に係る特許出願によって示される新従来技術：インライン展開による最適化を行わずに仮の実行プログラムを生成して実行し、その実行プログラムの実行によって得られる動的な（実際の）関数の呼出し回数やループの実行回数（ループ回数）を使用してインライン展開の要否を決定する。

【００１０】

【発明が解決しようとする課題】上述したように、旧従来技術では、インライン展開の要否の決定がプログラム全体を対象として行われており、関数のオブジェクトサイズ以外の要因（関数の呼出し回数等の要因）が考慮されていない。したがって、オブジェクトプログラムのサイズの増大等が生じてインライン展開が真の意味での最適化に寄与しない可能性がある（インライン展開によっ

てオブジェクトプログラムの実行効率を逆に落とすことがある）という欠点があった。

【００１１】また、「特開平３－９９３３０」に係る特許出願によって示される新従来技術では、インライン展開の要否を決定する際に新たに関数の参照回数という要因が考慮されているが、この参照回数は中間テキストに出現する参照回数であり実際の関数の呼出し回数とは異なるものである。したがって、ある関数に対する関数呼出しが中間テキスト中に何度も現れれば実際には一度もその関数が呼び出されなくてもインライン展開が行われてしまい（実行イメージに合わないインライン展開による最適化が行われることとなり）、無用にオブジェクトプログラムのサイズが大きくなってオブジェクトプログラムの実行効率の低下を招く可能性があるという欠点があった。

【００１２】さらに、「特開平１－１１８９３１」に係る特許出願によって示される新従来技術では、仮の実行プログラムを実際に実行した結果を使用してインライン展開の要否が決定されており、「仮の実行プログラムの生成および実行」という大掛かりな処理（一般的なコンパイラが持つ通常の最適化処理の範囲を逸脱する処理）が必要になる。したがって、実用的かつ効率的なインライン展開による最適化を行うことができないという欠点があった。

【００１３】本発明の目的は、上述の点に鑑み、関数呼出し毎にインライン展開の要否の決定を行うことを可能とし、実行イメージに合ったインライン展開を行うことができ、実用的かつ効率的なインライン展開を可能とする「インライン展開による最適化を行うコンパイラ」を提供することにある。

【００１４】

【課題を解決するための手段】本発明のインライン展開による最適化を行うコンパイラは、制御フロー解析手段の解析結果に基づいて制御フローを構成する各部分制御フローの実行回数を推測しその実行回数に基づいて各部分制御フローに対する重み付けを行う制御フロー重み付け手段と、この制御フロー重み付け手段による重み付けの結果を参照して各関数呼出しによって呼び出される関数のインライン展開の要否を決定しその決定を反映したオブジェクトプログラムを生成するオブジェクト生成手段とを有する。

【００１５】

【実施例】次に、本発明について図面を参照して詳細に説明する。

【００１６】図１は、本発明のインライン展開による最適化を行うコンパイラの第１の実施例（コンパイラ１０）の構成等を示すブロック図である。

【００１７】コンパイラ１０は、フロントエンド（字句解析部，構文解析部および意味解析部等の総称）１と、コンパイラ中間テキスト２と、制御フロー解析手段３

と、制御フロー情報付き中間テキスト４と、制御フロー重み付け手段５と、制御フロー重み情報付き中間テキスト６と、オブジェクト生成手段７とを含んで構成されている。また、コンパイラ１０は、ソースプログラム８を入力して、オブジェクトプログラム９を出力する。

【００１８】図２は、制御フロー重み付け手段５の詳細な構成等を示すブロック図である。制御フロー重み付け手段５は、制御フロー情報付き中間テキスト４を入力するループ検出部５１と、ループ回数判定部５２と、制御フロー重み情報付き中間テキスト６を出力する制御フロー重み付け部５３とを含んで構成されている。

【００１９】図３は、オブジェクト生成手段７の詳細な構成等を示すブロック図である。オブジェクト生成手段７は、関数オブジェクトサイズＳを出力する呼出し関数サイズ算出部７１と、制御フロー重み情報付き中間テキストによって示される部分制御フローの重み値Ｎと関数オブジェクトサイズＳとを入力してインライン展開判定値Ｊを出力するインライン展開判定値算出部７２と、インライン展開判定値Ｊを入力する関数インライン展開部７３とを含んで構成されている。なお、オブジェクト生成手段７は、インライン展開処理に関する機能（図３中の破線の枠内の機能）以外にもオブジェクトプログラム９の生成に関する多くの機能を有している。

【００２０】次に、このように構成された本実施例のインライン展開による最適化を行うコンパイラ（コンパイラ１０）の動作について説明する。

【００２１】フロントエンド１は、ソースプログラム８を入力し、従来技術と同様の方法で処理（字句解析，構文解析および意味解析等）を行い、コンパイラ中間テキスト２を出力する。

【００２２】制御フロー解析手段３は、コンパイラ中間テキスト２を入力し、従来技術と同様の方法で制御フロー解析を行い、制御フロー情報付き中間テキスト４（制御フロー解析の結果である制御フロー情報を有する中間テキスト）を出力する。

【００２３】制御フロー重み付け手段５内の各部は、以下に示すような処理を行う（図２参照）。

【００２４】ループ検出部５１は、制御フロー情報付き中間テキスト４を入力し、制御フロー情報によって示される制御フローからループ（ループ構造）を検出する。

【００２５】ループ回数判定部５２は、ループ検出部５１によって検出されたループの各々について、次のようにしてループ回数（ループの実行回数）を求める。

【００２６】○　処理対象のループを構成する条件分岐テキストを探し、その条件分岐テキストにおける条件が「変数Ｖ　比較演算　定数Ｃ１」という形であるか否かを判定する（この判定は、「静的にループ回数を求めることができるか否か」の判定となる）。

【００２７】①　当該条件が「変数Ｖ　比較演算　定数Ｃ１」という形である場合には、処理対象のループの入

口から制御フローをさかのぼり、変数Ｖに値を設定しているテキストを探す。

【００２８】②　そのテキストにおいて、変数Ｖに設定される値が定数Ｃ２であり、ループを制御するインダクション変数の増減値が定数Ｃ３である場合に、それらの定数Ｃ２およびＣ３と当該条件中の定数Ｃ１とを比較し、その比較に基づいてループ回数Ｌ（定数値）を求める。

【００２９】③　以上のようにして静的にループ回数Ｌを求めることができない場合（○の判定で当該条件が「変数Ｖ　比較演算　定数Ｃ１」という形でない場合，②において変数Ｖに設定される値が定数でない場合，または②においてループを制御するインダクション変数の増減値が定数でない場合）には、コンパイラ１０におけるループ回数のデフォルト値であるデフォルトループ回数αをループ回数Ｌとする。

【００３０】制御フロー重み付け部５３は、ループ回数判定部５２によって求められたループ回数Ｌを用いて、以下に示すようにして、制御フローを構成する各部分制御フローに対する重み付け（重み値Ｎの設定）を行う。なお、部分制御フローとは、最初のテキストから最後のテキストまでに分岐を示すテキストおよびラベルを示すテキストが存在せず連続して実行される制御フロー中の部分をいう。

【００３１】○　処理対象の部分制御フローがループに含まれない場合（ループの外側に存在する場合）には、その部分制御フローの重み値Ｎを１とする。

【００３２】①　処理対象の部分制御フローがネスト数ｎ（ｎは正整数）のループ群に含まれる場合には、その部分制御フローの重み値ＮをΠＬｉ（Ｌ１×Ｌ２×…×Ｌｎ。ｉ＝１，２，…，ｎでありＬｉはｉ層目のネストのループのループ回数）とする。

【００３３】すなわち、制御フロー重み付け部５３は、部分制御フローに関与するループ（部分制御フローを含むループ）についてのループ回数およびネスト数（部分制御フローがループに含まれるか否かという情報を含む）を使って当該部分制御フローの実行回数を推測し、ループをキーとする重み付け（重み値Ｎの設定）を当該部分制御フローに対して行う。

【００３４】制御フロー重み付け部５３は、上述のようにして設定した重み値Ｎを示す情報を有する中間テキストである制御フロー重み情報付き中間テキスト６を出力する。

【００３５】オブジェクト生成手段７は、従来通り、レジスタ割付け，コード生成および最適化等の処理を行う。オブジェクト生成手段７内の図３に示す各部は、このような処理の過程で、最適化処理の中の１つであるインライン展開処理を行う。このインライン展開処理は、最適化処理の過程で関数呼出しが検出された場合に起動され、以下に示すようにして行われる。

【００３６】呼出し関数サイズ算出部７１は、処理対象の関数呼出しによって呼び出される関数を探し、その関数のオブジェクトサイズＳを算出する。

【００３７】インライン展開判定値算出部７２は、当該関数呼出しを含む部分制御フローの重み値Ｎ（制御フロー重み付け手段５によって求められた値）と当該関数呼出しによって呼び出される関数のオブジェクトサイズＳとに基づいて、インライン展開判定値（インライン展開の要否を判定するための値）Ｊを算出する。ＮとＳとからＪを導く算出式は、ソースプログラム８の対象言語（ソースプログラム８を記述するプログラム言語）の特徴やオブジェクトプログラム９が実行されるターゲットマシンの特徴等に基づいて決定される。

【００３８】関数インライン展開部７３は、インライン展開判定値Ｊとコンパイラ１０におけるインライン展開要否判定用のデフォルト値であるインライン展開要否指標値βとを比較する。その比較で「Ｊ＞β」ならば、処理対象の関数呼出しを「インライン展開の対象とした方がよい関数呼出し」とみなして、その関数呼出しを関数（その関数呼出しによって呼び出される関数）の本体のオブジェクトテキストで置換する。

【００３９】オブジェクト生成手段７は、以上のようなインライン展開処理を行った上で、オブジェクトプログラム９を出力する。

【００４０】なお、本実施例に係るコンパイラ１０では、制御フロー重み付け手段５が請求項２記載の発明における制御フロー重み付け手段で実現される場合について述べた。しかし、制御フロー解析手段３の解析結果に基づいて何らかの方法で部分制御フローの実行回数を推測して各部分制御フローに対する重み付けを行うものであれば、請求項１記載の発明における制御フロー重み付け手段は本実施例における制御フロー重み付け手段５に限られるものではない。

【００４１】また、本実施例に係るコンパイラ１０では、オブジェクト生成手段７が請求項３記載の発明におけるオブジェクト生成手段で実現される場合について述べた。しかし、重み付けの結果を参照して各関数呼出しによって呼び出される関数のインライン展開の要否を決定するものであれば、請求項１記載の発明におけるオブジェクト生成手段は本実施例におけるオブジェクト生成手段７に限られるのものではない。

【００４２】図４は、本発明のインライン展開による最適化を行うコンパイラの第２の実施例（コンパイラ２０）の構成等を示すブロック図である。

【００４３】コンパイラ２０は、フロントエンド１と、コンパイラ中間テキスト２と、制御フロー解析手段３と、制御フロー情報付き中間テキスト４と、制御フロー重み付け手段５と、制御フロー重み情報付き中間テキスト６と、オブジェクト生成手段７と、オプション情報１１とを含んで構成されている。また、コンパイラ２０

は、ソースプログラム８を入力して、オブジェクトプログラム９を出力する。すなわち、本実施例（第２の実施例）に係るコンパイラ２０は、図１に示す第１の実施例に係るコンパイラ１０に対してオプション情報１１が追加された構成になっている。したがって、オプション情報１１に関する処理以外は、本実施例における処理と第１の実施例における処理とは同様である。この意味で、図１と図４とで対応する構成要素（コンパイラ１０および２０自身は除く）については同一の符号を用いて示している。

【００４４】なお、図２および図３は、本実施例の制御フロー重み付け手段５およびオブジェクト生成手段７の詳細な構成等を示すブロック図でもある。

【００４５】次に、このように構成された本実施例のインライン展開による最適化を行うコンパイラ（コンパイラ２０）の動作について説明する。基本的な動作の流れはコンパイラ１０の動作の流れと同様であるので、ここではコンパイラ１０の動作と異なる動作についてのみ列挙する。

【００４６】○　フロントエンド１は、ソースプログラム８を入力してコンパイラ中間テキスト２を出力する際に、コンパイラ２０の起動時にユーザによって与えられたオプション（指定ループ回数および指定指標値）を有するオプション情報１１を生成する。

【００４７】①　制御フロー重み付け手段５内のループ回数判定部５２は、静的にループ回数を決めることができない場合に、オプション情報１１に基づいてデフォルトループ回数αを求める。すなわち、ユーザによって指定された指定ループ回数をデフォルトループ回数αとして使用する。

【００４８】②　オブジェクト生成手段７内の関数インライン展開部７３は、オプション情報１１に基づいてインライン展開要否指標値βを取得して、そのインライン展開要否指標値βとインライン展開判定値Jとの比較を行う。すなわち、ユーザによって指定された指定指標値をインライン展開要否指標値βとして使用する。

【００４９】このように、本実施例では、ユーザはコンパイル時にソースプログラム８の特性やユーザの要求等に応じてデフォルトループ回数やインライン展開要否指標値を指定することができる。

【００５０】なお、ユーザによってオプションとして与えられる情報は、本実施例のように指定ループ回数および指定指標値の両方である場合だけではなく、指定ループ回数および指定指標値のいずれか一方だけである場合も許容される。

【００５１】

【発明の効果】以上説明したように本発明は、旧従来技術においてプログラム全体を対象として行われていたインライン展開の要否の決定を関数呼出し毎に行うことにより、より詳細な情報によって真の意味での最適化（インライン展開による最適化）の実現の可能性が高くなるという効果がある。

【００５２】また、関数呼出し毎にインライン展開を行うか否かを決定する要因として制御フロー解析に基づいて求められる部分制御フローの実行回数を使っているので、実際の実行イメージに合わない最適化（「特開平３－９９３３０」に係る特許出願によって示される中間テキスト上の関数の参照回数を使う最適化等）や通常のコンパイラの最適化処理の範囲を逸脱する処理が必要になる最適化（「特開平１－１１８９３１」に係る特許出願によって示される大掛かりな仕組みを用いる最適化等）を避けることができ、実行イメージに適合するとともに実用的かつ効率的なものとなる最適化（インライン展開による最適化）を行うことができるという効果がある。

【００５３】さらに、インライン展開の要否のキーとなる値をユーザがオプションによって指定することを可能とすることにより（請求項４および請求項５参照）、ソースプログラムの特性やユーザの要求等に応じたインライン展開による最適化を行うことができるようになるという効果がある。

【図面の簡単な説明】

【図１】本発明の第１の実施例の構成等を示すブロック図である。

【図２】図１および図４中の制御フロー重み付け手段の詳細な構成等を示すブロック図である。

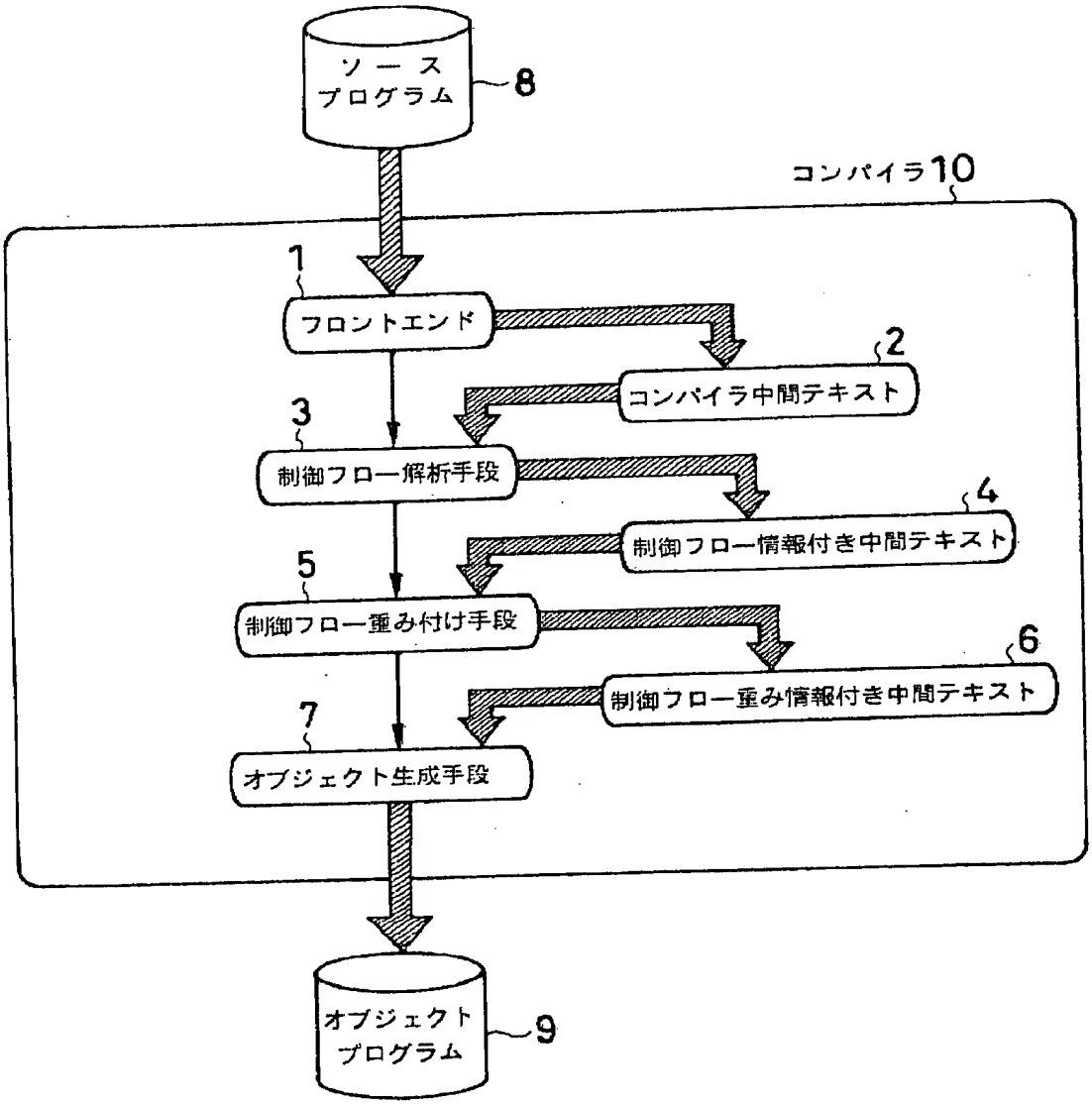
【図３】図１および図４中のオブジェクト生成手段の詳細な構成等を示すブロック図である。

【図４】本発明の第２の実施例の構成等を示すブロック図である。

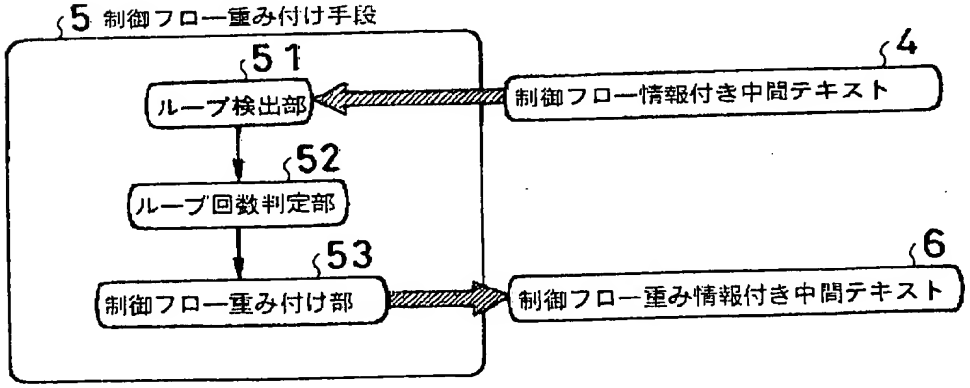
【符号の説明】

１　フロントエンド
２　コンパイラ中間テキスト
３　制御フロー解析手段
４　制御フロー情報付き中間テキスト
５　制御フロー重み付け手段
６　制御フロー重み情報付き中間テキスト
７　オブジェクト生成手段
８　ソースプログラム
９　オブジェクトプログラム
１０，２０　コンパイラ
１１　オプション情報

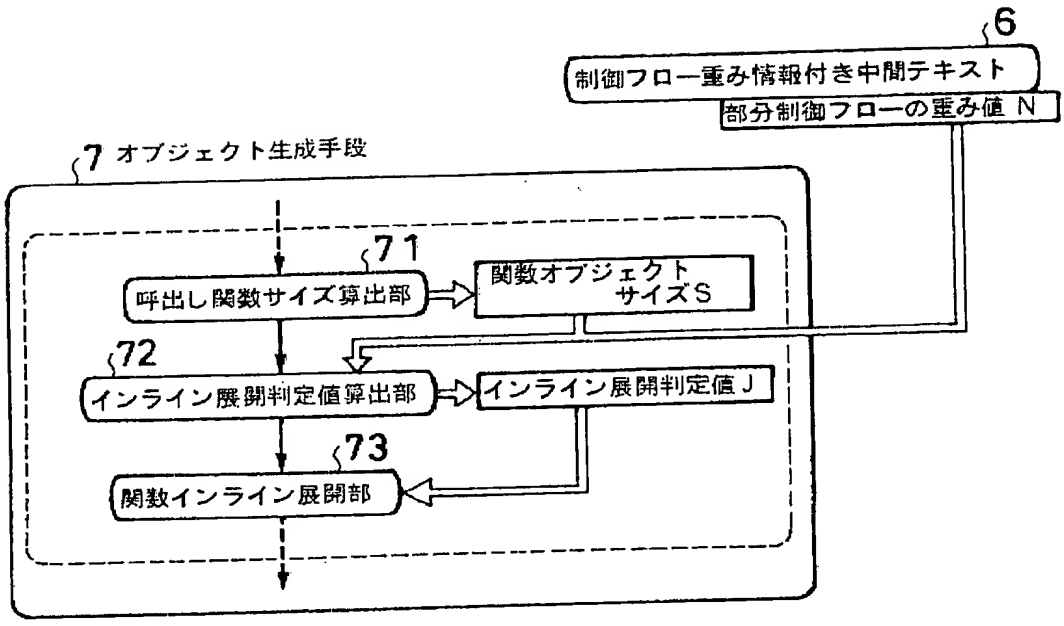
【図１】


ソース
プログラム
8
コンパイラ10
1
フロントエンド
2
コンパイラ中間テキスト
3
制御フロー解析手段
4
制御フロー情報付き中間テキスト
5
制御フロー重み付け手段
6
制御フロー重み情報付き中間テキスト
7
オブジェクト生成手段
オブジェクト
プログラム
9

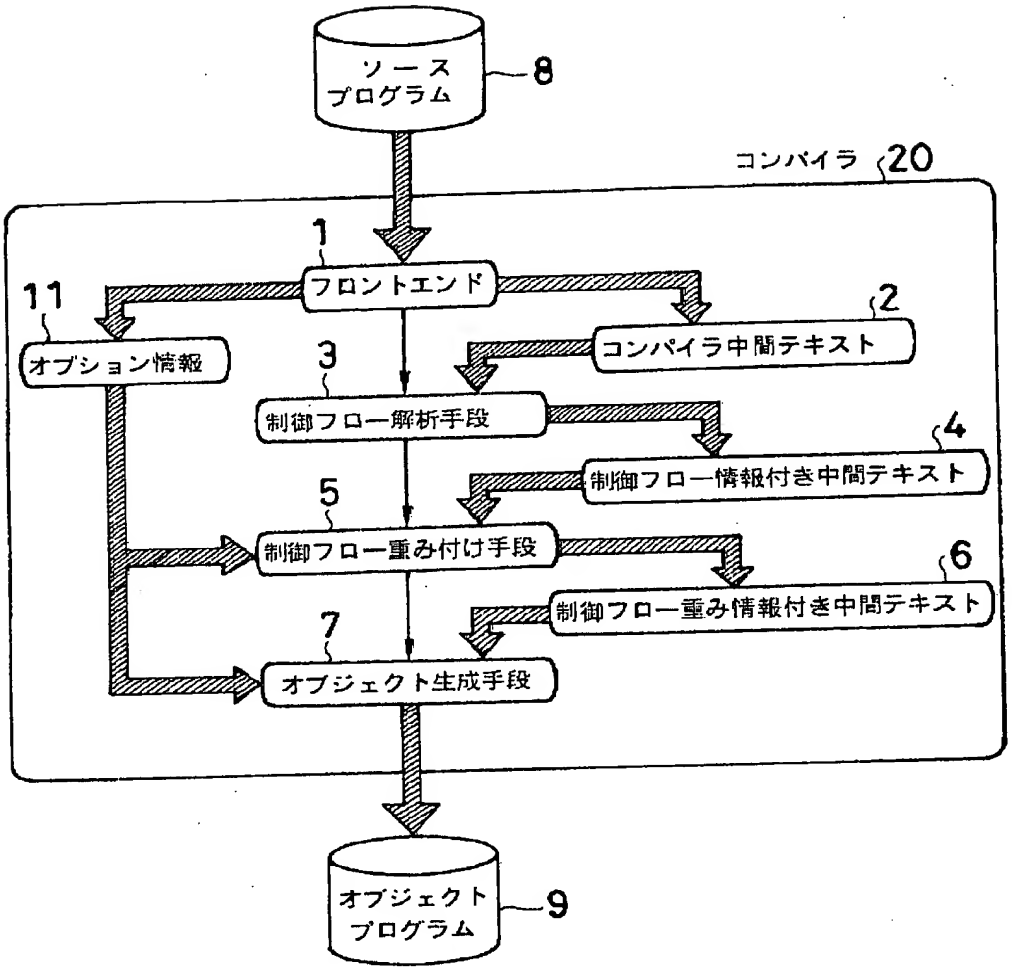
【図２】


5 制御フロー重み付け手段
51
ループ検出部
4
制御フロー情報付き中間テキスト
52
ループ回数判定部
53
制御フロー重み付け部
6
制御フロー重み情報付き中間テキスト

【図３】


6
制御フロー重み情報付き中間テキスト
部分制御フローの重み値 N
7 オブジェクト生成手段
71
呼出し関数サイズ算出部
関数オブジェクト
サイズS
72
インライン展開判定値算出部
インライン展開判定値 J
73
関数インライン展開部

【図４】


ソース
プログラム
8
コンパイラ
20
1
フロントエンド
11
オプション情報
2
コンパイラ中間テキスト
3
制御フロー解析手段
4
制御フロー情報付き中間テキスト
5
制御フロー重み付け手段
6
制御フロー重み情報付き中間テキスト
7
オブジェクト生成手段
オブジェクト
プログラム
9